Snakes

rs@toprllc.com

Assignments

July 6, 2020

Snakes - Further explorations in GUI

Snakes is a simple game that has surprising challenges in implementation. In this projectlet, we will investigate - techniques for separating the algorithms from their presentation and user interaction. In addition, this affords us the opportunity to compare several GUI libraries that differ in their approach.

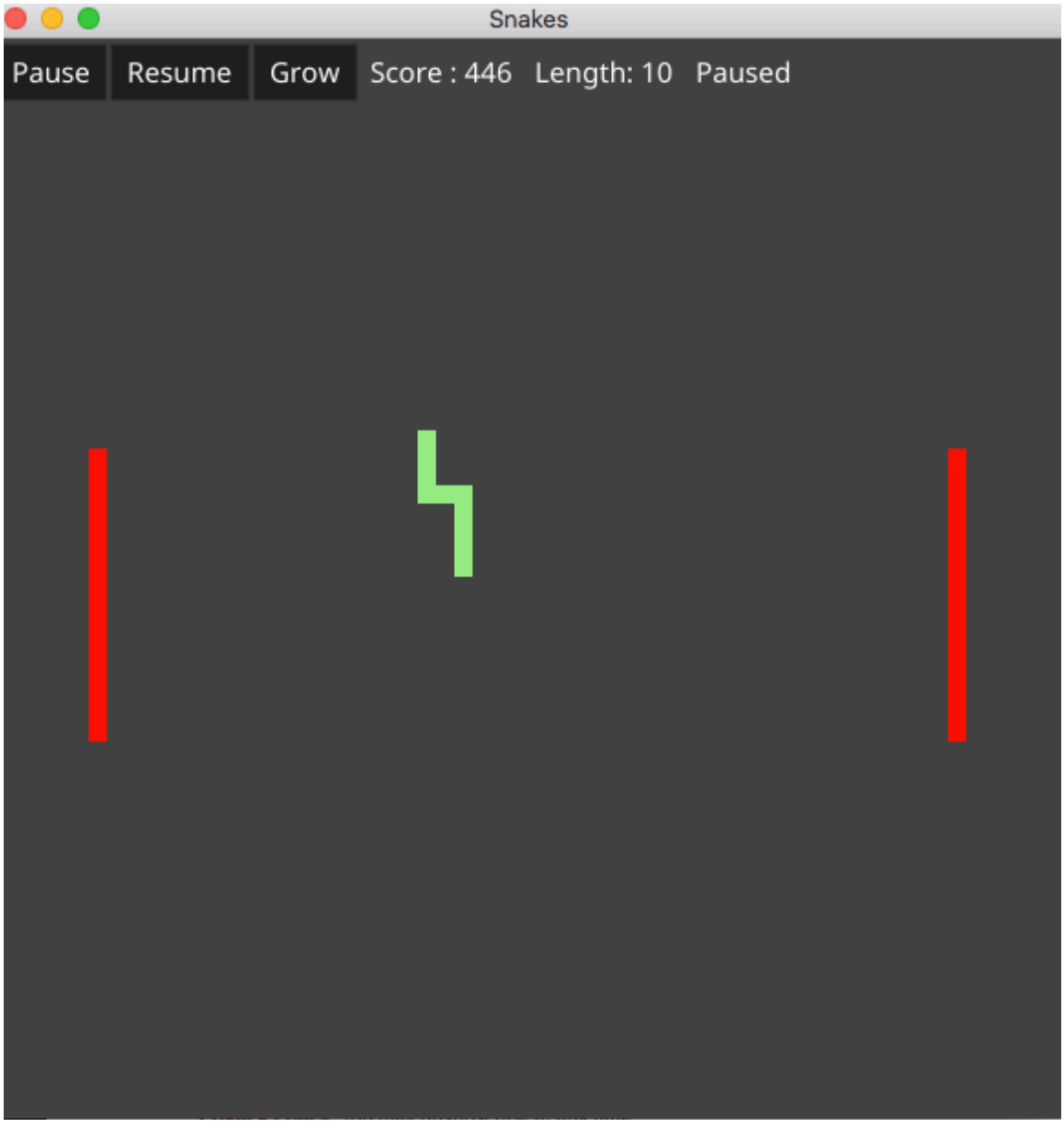
Brief Description

The game uses a rectangular board of squares with some sets of contiguous squares being barriers. The snake itself is a set of contiguous squares with a head and a direction of travel - North, South, East or West. Each move, the snake moves 1 square unless it is confronted by a barrier or the border of the board. The user has the option to change the direction of travel of the snake. The snake is also able to grow on request. The Figure 1 is an illustration. The red blocks represent barriers and the green blocks are the snakes.

Design Challenges

A. The GUI library. Availability of a suitably accessible library cannot always be assumed. Depending on the language, desired target platforms, the choices may be a bit limited. For example https://www.gtk.org is a cross platform library but if your target happens to be a bare metal embedded system, this may not be the right choice.

B. Every library has a different approach to handling user input. For instance handling



1. **SNAKE PAUSED - EXAMPLE USING FYNE**

key press events is somewhat different between fyne and gtk. A design that distances itself from actual keycodes will tend to grow easily and migrate to other platforms.

C. Application separation from the user interface will often be critical for the long term viability of the product. A better algorithm may be invented and for preserving the customer experience, the UI might need to be preserved. On the other hand, as newer delivery platforms e.g. tablets become prevalent the interactions may have to change.
D. Performance - An obvious consideration, graphical user interfaces are more sensitive to the resources available e,g. CPU power, RAM etc, than command line utilities.
E. Testing - Software of even moderate complexity becomes incredibly hard to test when User Interfaces are involved. The more options supported by the program, the more futile it is to attempt the quality control manually without automation.

Solution Architecture

In this solution, the language go is chosen. This affords us the choice of several GUI libraries: fyne and gotk3 which in turn is a binding to the gtk3 library. These choices lead to being able to support Linux, Mac OS and Windows platforms. An implementation is available from:

https://bitbucket.org/ToprLLC/snakes/src/master/

Execution:

$ snakes [vertical|horizontal] [barriers]

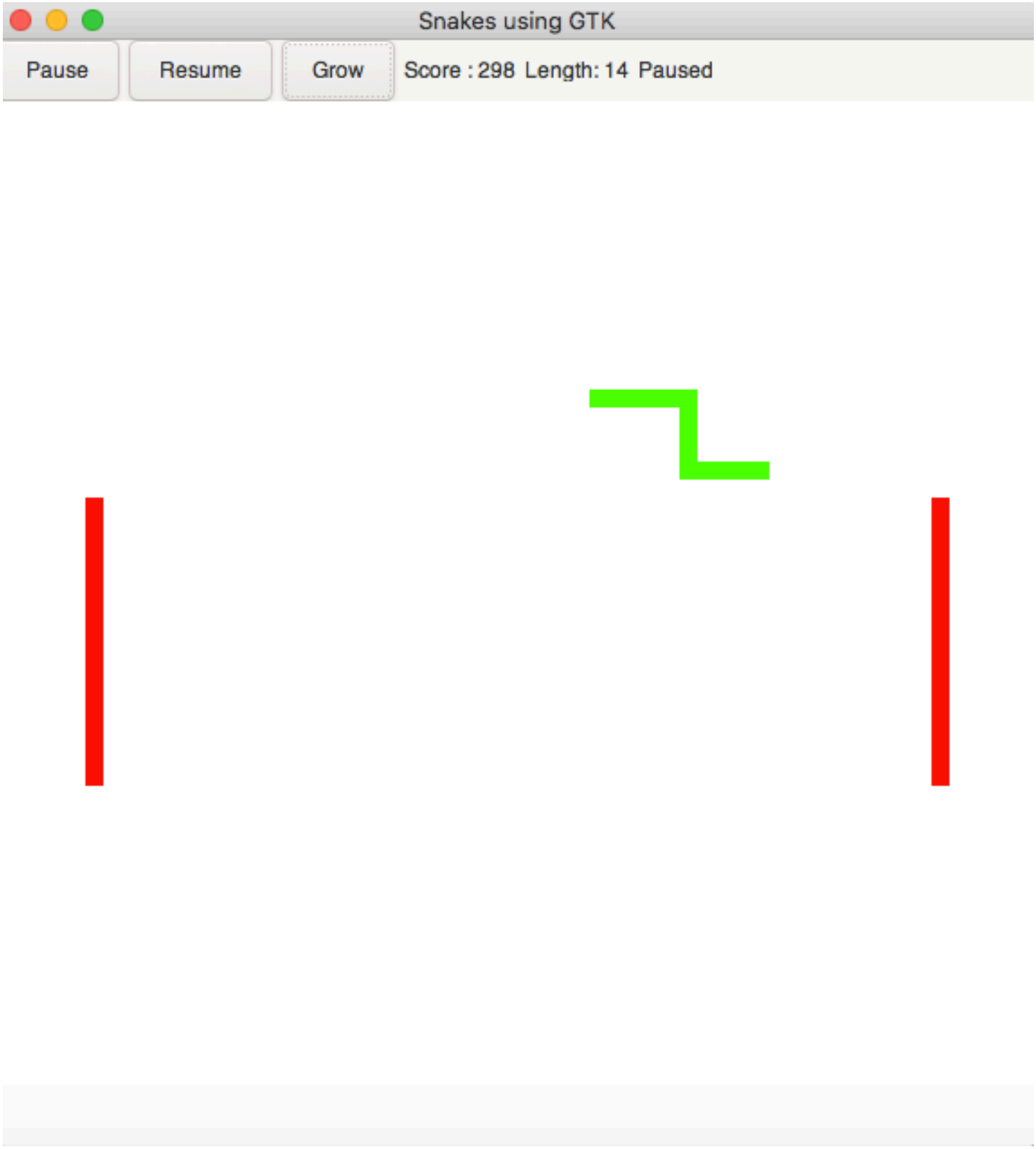
barriers - asks 2 barriers to be created in an orientation opposing the snake orientation

vertical|horizontal - asks the snake to be created with the initial orientation as specified. It will be located with the head at the center of the board.

Different executables are produced - one for each graphical library. e.g. **bin/fyne/snakes** and **bin/gtk3/snakes**.

References

I. fyne - https://fyne.io
II. gtk - https://github.com/gotk3/gotk3



2 EXAMPLE USING GTK3